Reinforcement Learning: An Introduction
Solutions: Chapter 1

Mrinank Sharma

March 24, 2020

Exercise 1.1: Self-Play

Since the RL algorithm is now playing against itself, the opponent is changing. The optimal strategy ought to be stable - if both players are playing optimally, they have no incentive to change their moves, so if the algorithm reachs this stage, it ought to be stable. However, there is no guarantee that this is reached. Perhaps the policy that the agent learns could form loops and never converge.

Exercise 1.2: Symmetries

The learning process proposed in the book uses value functions for each state. If certain positions are effectively identical, we should treat their value as being the same (with value functions, we can simply label the states are being equivalent). This should make the learning process faster i.e., we need fewer iterations (or sample games) to learn the correct value functions.

However, states are only effectively identical if the opponent treats them as such. If the opponent does not realise the symmetry between states and thus takes non-corresponding actions in these states, our RL agent should try to take advantage of this. In this setting, these symmetric positions do not have the same value because the opponent players differently in these states.

Exercise 1.3: Greedy Play

Recall that in the algorithm suggested by the book, we initialise the value functions for all of the states and then perform some sort of ϵ-greedy policy. If the player is *purely greedy* i.e., never takes any exploratory actions, there is no guarantee that the values for all states would eventually reach the 'correct' numbers, being that the play will be worse (or in the best case the same depending on the environment and initialisation).

If we magically already knew that correct values for each of the states, there would be no problem.

Exercise 1.4: Learning from Exploration

In the case where we update after all moves, even after exploratory moves, we would converge to the values *under the ϵ-greedy policy* i.e., probability of winning assuming that we are mostly greedy and sometimes exploratory.

In the case given, we converge to the probability of winning assuming that we always take the best move.

If we care about wins during the training phase, we'll want to learn the probability of winning assuming the exploratory policy as there could be moves which are identical under optimal play but different under sub-optimal play. But it seems that we wouldn't deploy with this exploratory policy, and we might not care about performance during the learning phase, so its better to learn the optimal win probabilities.

Exercise 1.5: Other Improvemenst

At the moment, the exploration policy is purely random. We could instead use a smarter exploration policy, such as choosing to explore the trajectories through the game tree which we haven't seen before (or using some UCB or similar policy instead).

We could also initialise our value functions better by using our knowledge of Tic-tac-toe i.e., we know there will be board moves with low probabilities of winning and those with high probabilities of winning. Initialising these better should also speed up training.